

Heuristic with elements of tabu search for Truck and Trailer Routing Problem

Ivan S. Grechikhin

Abstract Vehicle Routing Problem is a well-known problem in logistics and transportation, and the variety of such problems is explained by the fact that it occurs in many real-life situations. It is an NP-hard combinatorial optimization problem and finding an exact optimal solution is practically impossible. In this work, Site-Dependent Truck and Trailer Routing Problem with hard and soft Time Windows and Split Deliveries is considered (SDTTRPTWSD). In this article, we develop a heuristic with the elements of Tabu Search for solving SDTTRPTWSD. The heuristic uses the concept of neighborhoods and visits infeasible solutions during the search. A greedy heuristic is applied to construct an initial solution.

Key words: Truck and Trailer Routing Problem, Site-Dependent, Soft Time Windows, Split Deliveries, Tabu search

1 Introduction

Vehicle Routing Problem is a well-known problem in combinatorial optimisation and integer programming. The problem can be described as follows: there is a set of customers, where each customer has a demand, there is a set of vehicles, which may serve the demand of customers. Using the information on the distances and costs of travelling between each pair of customers, the goal is to find the solution with minimal total cost. This paper considers one version of the problem, which is called Truck and Trailer Routing Problem (TTRP). The problem in consideration is a real-life problem, and contains a big number of constraints.

⁰ National Research University Higher School of Economics. Laboratory of Algorithms and Technologies for Network Analysis. Probationer; E-mail: igrechikhin@hse.ru

Truck and trailer routing problem has two sets of customers: truck-customers and trailer-customers. Every vehicle, then, consists of a truck and a trailer of some capacities (sometimes trailer capacity is zero, which means the vehicle does not have a trailer). Truck-customers can not be served by a vehicle with a trailer. It means that the vehicle should not have trailer from the start, or the trailer should be left at some other place before visiting a truck-customer. This requirement is explained by the fact that there may be small stores, that do not have place for vehicle with its trailer. A vehicle with a trailer has a possibility to leave the trailer at a transshipment location, which is basically a special place to leave trailers. Another opportunity is to leave trailer at a previous trailer-customer: in this case the trailer may be unloaded at the trailer-customer and, at the same time, the truck goes to a truck-customer and serves it in parallel with the trailer-customer. Such rules create a necessity to organize load transfer - the operation, where goods are transferred from truck to trailer or vice versa. This may happen, because, for example, the total weight of goods for truck-customers in one route is more than the capacity of the truck.

In the considered problem, the Heterogeneous Fleet of vehicles is present (HFT-TRP). This problem differs from homogeneous fleet TTRP, where all vehicles are the same: they have the same fixed costs and capacities. HFTTRP has a set of vehicles with different capacities and fixed costs, which makes the problem even more difficult. Additionally, every customer may have its own preferences on types of vehicles to serve the customer. In this case the problem is called the Site Dependent TTRP (SDTTRP) and there are some developed heuristics for solving such problems sometimes with additional elaborations in constraints .

Another real-life constraints are hard and soft time windows and split-deliveries. Time windows are periods of time, when the delivery is acceptable (hard time windows) and the constraint should be satisfied in the majority of routes (soft time window). Split-deliveries are such deliveries, when there is a possibility to serve one customer with more than one vehicle. The problem in this form is considered in Batsyn & Ponomarenko (2014) and Batsyn & Ponomarenko (2015). These papers suggested greedy heuristic for the problem. In this article, this heuristic is developed in another way with an addition of new heuristic with elements of tabu-search. The greedy heuristic is altered so there are possible operations of both insertion and deletion from the route. After the greedy heuristic, the obtained solution is reconstructed with new heuristic with tabu-search elements.

Greedy heuristic constructs the solution iteratively, until there are no unserved customers. For every route, the algorithm randomly chooses one of the farthest customers to be the first customer added to the route. Then, other customers are tried as candidates to the route. The solution has a constraint on the number of split-deliveries and delays (violations of soft time window). For every route, the possibility of a split-delivery and the number of delays is chosen randomly. The route may have only one new split-delivery, and the probability is determined by the fraction of the current allowed split-deliveries to the estimated number of split-deliveries. The number of delays is defined by the fraction of the current allowed soft win-

dow violations to the estimated number of soft window violations, however, every route may have different number of delays. The algorithm determines the allowed number of delays before constructing the route. After the solution is obtained, the heuristic with tabu-search elements tries to move customers between the routes to derive better solution. The algorithm uses set of changing parameters, which define “tabu neighbourhood” - the algorithm looks through the infeasible solutions. The degree of infeasibility is determined by the parameters - the number of allowed delays over limit, the number of routes with violated capacity and allowed cost change of the route.

2 Heuristic algorithm

The following parameters are used in the pseudo-code of the algorithm:

n - the number of customers

V - the set of all customers

K - the set of all vehicles

K_i - the set of vehicles, which can serve customer i

Q_k - the current remaining capacity of vehicle k

q_i - the current remaining demand of the customer i

v_R - the number of soft time window violations in route R

R - the current route

S - the current solution

S^* - the best solution found so far

v - the number of permitted soft window violations

w - the current remaining number of permitted soft time window violations

U - the set of all customers sorted the most expensive (farthest) customer first

C - the cost of current insertion

corridor - the allowed level of violations

CurrentState - the current state of second heuristic

closeness - the distance between customers to perform move

CVset - the set of routes, where the capacity of vehicles is violated

The first important function of the whole algorithm is initial greedy heuristic, which constructs initial solution (Algorithms 1 and 2). The function works so that the solution will be necessarily constructed, but its cost may not be satisfactory. First, the algorithm sorts all customers by the distance from the depot (or, by the cost of direct travel from depot, which is the same) so the first customer in U is the farthest. Then, the process of solution construction begins. Routes of the solution are constructed in cycle, until there are unserved customers. For every route, the algorithm chooses one of the farthest customers, after that the vehicle is determined for the route. Also, function `BASICROUTE(k)` creates the route with one chosen customer.

The function `FINDNUMBEROFVIOLATIONS(i, k, w)` determines maximal possible number of soft window violations for the current route. The function uses the rela-

Algorithm 1 Initial Greedy heuristic Part 1

```

1: function INITIALGREEDYHEURISTIC
2:    $\triangleright$  Creates one initial feasible solution
3:    $U \leftarrow V$   $\triangleright$  sorting customers so that  $U_1$  has maximal  $c_{0i}^{kl}$ 
4:    $S \leftarrow \emptyset$ 
5:   while  $U \neq \emptyset$  do
6:      $w = v$ 
7:      $i \leftarrow \text{RANDOM}(U_1, \dots, U_\mu)$   $\triangleright$  choose from the  $\mu$  most expensive
8:      $k \leftarrow \text{CHOOSEVEHICLE}(i, [q_j], [Q_k])$ 
9:      $R \leftarrow \text{BASICROUTE}(k)$ 
10:     $violNumber \leftarrow \text{FINDNUMBEROFVIOLATIONS}(i, k, w)$ 
11:     $R_{viol} \leftarrow \emptyset, R_{clear} \leftarrow \emptyset$ 
12:     $C_{viol} \leftarrow \infty, C_{clear} \leftarrow \infty$ 
13:     $ID_{viol} \leftarrow 0, ID_{clear} \leftarrow 0$ 
14:    for  $j \in U$  do
15:      if  $k \notin K_j$  then
16:        continue
17:      end if
18:       $mayViolate \leftarrow true$ 
19:       $C'_{viol} \leftarrow \text{GETINSERTIONCOST}(j, R, mayViolate, q_j, Q_k, violNumber)$ 
20:       $mayViolate \leftarrow false$ 
21:       $C'_{clear} \leftarrow \text{GETINSERTIONCOST}(j, R, mayViolate, q_j, Q_k, violNumber)$ 
22:       $\triangleright$  There are two possible insertions, with violation or without
23:      if  $C'_{viol} < C_{viol}$  then
24:         $C_{viol} \leftarrow C'_{viol}$ 
25:         $R_{viol} \leftarrow \text{INSERTCUSTOMER}(j, R, true, q_i, Q_k)$ 
26:         $ID_{viol} \leftarrow j$ 
27:      end if
28:      if  $C'_{clear} < C_{clear}$  then
29:         $C_{clear} \leftarrow C'_{clear}$ 
30:         $R_{clear} \leftarrow \text{INSERTCUSTOMER}(j, R, false, q_i, Q_k)$ 
31:         $ID_{clear} \leftarrow j$ 
32:      end if

```

tion of current remaining soft window violations to the estimated number of remaining soft window violations and increases the number of allowed violations until the random generator returns numbers less than this relation. After that, the algorithm tries to insert all other customers in the route R , however, the algorithm does the insertion in two ways - allowing the violation of soft time window and forbidding the violation. If the number of soft window violations exceeds the allowed number, the route is forbidden. From obtained routes, there is chosen the best. Step by step the algorithm inserts customers until there are no possible insertions.

Algorithm 2 Initial Greedy heuristic Part 2

```

33:      if  $R_{viol} = null$  and  $R_{clear} = null$  then
34:           $S \leftarrow S \cup \{R\}$ 
35:           $Q_k \leftarrow 0$ 
36:           $w \leftarrow w - v_R$ 
37:          break
38:      else if  $R_{clear} = null$  then
39:           $R \leftarrow R_{viol}$ 
40:           $U \leftarrow U / \{ID_{viol}\}$ 
41:      else if  $R_{viol} = null$  then
42:           $R \leftarrow R_{clear}$ 
43:           $U \leftarrow U / \{ID_{clear}\}$ 
44:      else
45:          if  $C_{viol} > C_{clear}$  then
46:               $R = R_{viol}$ 
47:               $U \leftarrow U / \{ID_{viol}\}$ 
48:          else
49:               $R = R_{clear}$ 
50:               $U \leftarrow U / \{ID_{clear}\}$ 
51:          end if
52:      end if
53:  end for
54:  end while
55: end function

```

The whole idea of the greedy algorithm is based on Batsyn & Ponomarenko (2014) and Batsyn & Ponomarenko (2015)

Second important function is the second heuristic with elements of tabu search (Algorithm 3). Its goal is to take initial solution S and improve it by performing simple moves. The algorithm makes steps and at each step there is a possible move happens. The variety of possible moves depends on the *corridor* and *closeness* parameters. Also, there is *CurrentState* of the algorithm, which tracks successes, changes in the current best and some other parameters. From time to time, the algorithm tries to obtain feasible solution from current solution. The algorithm also may change *corridor* depending on *CurrentState* of the heuristic or even stop it in order to get new initial solution and start the procedure again.

At every step of the second heuristic (Algorithm 4), first, the customer is chosen randomly from one of the route of the current solution. After that, the customer is tried to be inserted in other routes in such way that the adjacent customer is close - the time of travel is less than *closeness* parameter. Variables *places* contains all such places of insertion in the route R . After finding the best move by the cost this move may be performed if it does not violate too many constraints.

Algorithm 3 Heuristic with Tabu Search

```

1: function TABUSEARCHHEURISTIC( $S, corridor, closeness, CurrentState$ )
2:    $S^* \leftarrow S$ 
3:   repeat
4:      $success \leftarrow \text{HEURISTICSTEP}(S, corridor, closeness)$ 
5:      $\text{CHANGESTATEFORTABUSTEPSUCCESS}(CurrentState, success)$ 
6:     if SHOULD OBTAIN FEASIBLE SOLUTION( $CurrentState$ ) then
7:       ▷ Recovery procedures work here
8:       ROUTE OPTIMIZATION( $S$ )
9:       RECOVER CAPACITY VIOLATIONS( $S$ )
10:      FINALIZE ROUTE TIMES( $S$ )
11:      RECOVER SOFT WINDOW VIOLATIONS( $S$ )
12:      if  $\text{COST}(S) < \text{COST}(S^*)$  then
13:         $S^* \leftarrow S$ 
14:         $\text{CHANGESTATEFORCHANGEINBEST}(CurrentState)$ 
15:      end if
16:    end if
17:     $\text{CHANGECORRIDOR}(corridor, CurrentState)$ 
18:  until STOPPING CONDITION( $CurrentState$ )
19: end function

```

Finally, when the second heuristic tries to obtain the feasible solution from current infeasible, the recovery procedure takes place (Algorithm 3). Basically, the whole solution is likely to be in infeasible region because of moves. In that case, the algorithm needs to decrease the number of soft time window violations and recover over-capacitated routes to be under constraints. The recovery procedures start with route optimization - it creates some free space inside routes in order to recover solution more efficiently. After that, the algorithm recovers capacities of routes. Next step is finalization of times - the procedure goes through every route and compacts the time of the route. The last step is recovering soft time window violations.

The algorithm of capacity constraints recovery is described in Algorithm 5. There are two parts in this algorithm. First part of the algorithm repeatedly tries to take customers from over-capacitated routes and insert them in other routes without capacity violations. If there is no such move possible and there are over-capacitated routes left, the second part of the algorithm creates new routes with customers from over-capacitated routes. At the end of the procedure all routes have total demand less or equal to the capacity of the vehicle of the route.

3 Computational results

Experiments were performed for seven experimental days, for which the good results of greedy heuristic are known. The column Greedy Heuristic Results, contains

Algorithm 4 Heuristic Step Algorithm

```

1: function HEURISTICSTEP( $S, corridor, closeness$ )
2:    $R_i, i, costOfDeletion \leftarrow \text{CHOOSERANDOMCUSTOMER}(S)$ 
3:    $\triangleright i$  is deleted customer, the algorithm also needs the cost of deletion of this
      customer from its current route
4:    $bestCost \leftarrow \infty$ 
5:    $bestRoute \leftarrow \emptyset$ 
6:   for all  $R \in S$  do
7:      $places \leftarrow \text{FINDPLACESFORINSERTION}(R, closeness)$ 
8:     for all  $place \in places$  do
9:        $R^* \leftarrow \text{ADDCUSTOMER}(R, place)$ 
10:       $cost \leftarrow \text{FINDMOVECOST}(costOfDeletion, R, R^*, corridor, CurrentState)$ 
11:      if  $cost < bestCost$  then
12:         $bestCost = cost$ 
13:         $bestRoute = R$ 
14:      end if
15:    end for
16:  end for
17:   $success \leftarrow \text{ALLOWMOVE}(bestCost, corridor)$ 
18:  if  $success$  then
19:     $R_i \leftarrow R_i / i$ 
20:     $R \leftarrow R \cup i$ 
21:     $\text{CHANGECURRENTVIOLATIONS}(CurrentState, S)$ 
22:  end if
23: end function

```

the value of objective function obtained by the greedy heuristic for this day (Batsyn & Ponomarenko, 2015). The third column shows the results of heuristic with tabu search elements for the experimental days. The second heuristic worked for 3 hours for every experimental day. All experiments were conducted on Intel Xeon X5675 machine, with base processor frequency 3.06 GHz and 64 GB of memory.

Table 1: Computational results

Day	Greedy Heuristic Results	Tabu Search Heuristic Results	Improvement
Day 1	1200000	1155000	-4%
Day 2	1100000	1100000	0%
Day 3	1160000	1100000	-5%
Day 4	1200000	1140000	-5%
Day 5	1245000	1220000	-2%
Day 6	1235000	1225000	-1%
Day 7	1275000	1175000	-8%

Algorithm 5 Recovery Capacity Violations Procedure Part 1

```

1: function RECOVERCAPACITYVIOLATIONS( $S$ )
2:    $CVset \leftarrow \text{FINDROUTESWITHCAPACITYVIOLATIONS}(S)$ 
3:   for all  $R \in CVset$  do
4:      $bestCost \leftarrow \infty$ 
5:      $bestCustomer \leftarrow -1$ 
6:      $bestRouteFrom \leftarrow \emptyset$ 
7:      $bestRouteTo \leftarrow \emptyset$ 
8:     for all  $i \in R$  do
9:       for all  $R^c \notin CVset$  do
10:         $cost, R^*, R^{c*} = \text{FINDCOSTOFMOVE}(R, i, R^c)$ 
11:        if  $cost < bestCost$  then
12:           $bestCost \leftarrow cost$ 
13:           $bestCustomer \leftarrow i$ 
14:           $bestRouteFrom \leftarrow R^*$ 
15:           $bestRouteTo \leftarrow R^{c*}$ 
16:        end if
17:      end for
18:    end for
19:    if  $bestCost \neq \infty$  then
20:       $\text{REPLACEROUTES}(S, bestRouteFrom, bestRouteTo)$ 
21:    end if
22:  end for
23:   $CVset = \text{FINDROUTESWITHCAPACITYVIOLATIONS}(S)$ 
24:  while  $CVset \neq \emptyset$  do
25:     $R_{cap} \leftarrow \text{CHOOSERANDOMROUTE}(CVset)$ 
26:     $i \leftarrow \text{CHOOSECUSTOMERTORECOVERCAPACITY}(R_{cap})$ 
27:     $CVset \leftarrow CVset / R_{cap}$ 
28:     $success \leftarrow (CVset \neq \emptyset)$ 
29:    while  $R \in CVset$  do
30:       $bestCost \leftarrow \infty$ 
31:       $bestCustomer \leftarrow -1$ 
32:       $bestRouteFrom \leftarrow \emptyset$ 
33:       $bestRouteTo \leftarrow \emptyset$ 
34:      for all  $i \in R$  do
35:        for all  $R^c \notin CVset$  do
36:           $cost, R^*, R^{c*} = \text{FINDCOSTOFMOVE}(R, i, R^c)$ 
37:          if  $cost < bestCost$  then
38:             $bestCost \leftarrow cost$ 
39:             $bestCustomer \leftarrow i$ 
40:             $bestRouteFrom \leftarrow R^*$ 
41:             $bestRouteTo \leftarrow R^{c*}$ 
42:          end if
43:        end for
44:      end for
45:    end while
46:  end while
47: end function

```

4 Conclusion

In this paper new heuristic was developed for the Site-Dependent Truck and Trailer Routing Problem with Time Windows and Split Deliveries. The heuristic uses a greedy approach for the initial solution construction and then employs elements of local search and tabu search to improve the initial solution. The obtained results are promising as they show improvement in most cases.

The following work should be directed to the improvement of the speed of the algorithm and to guarantee the best possible results as well. One of the way to improve the algorithm is to use new neighborhood - swap neighborhood, where two customers from different routes can be swapped. Also, there are more constraints that can be relaxed, such as time windows and split deliveries.

5 Acknowledgments

The author is supported by LATNA Laboratory, NRU HSE, RF government grant, ag. 11.G34.31.0057.

6 References

1. Batsyn, M., & Ponomarenko, A. (2014). *Heuristic for a Real-life Truck and Trailer Routing Problem*. *Procedia Computer Science*, 31, 778-792. doi:10.1016/j.procs.2014.05.328
2. Batsyn, M., & Ponomarenko, A. (2015). *Heuristic for Site-Dependent Truck and Trailer Routing Problem with Soft and Hard Time Windows and Split Deliveries*. *Lecture Notes in Computer Science Machine Learning, Optimization, and Big Data*, 65-79. doi:10.1007/978-3-319-27926-8_7